정렬 알고리즘 코드테스트 정리

학습 대상: 백준 + 프로그래머스 정렬 문제
학습 목적: 실전 코딩테스트 문제 유형별 정렬 알고리즘, STL 사용법, 입력/출력 최적화, 그리고 문제 접근 방식의 차이점 분석

1. STL 자료구조 및 기본 이해

정렬 알고리즘 학습과 함께 반드시 병행해야 할 것이 **STL 자료구조의 이해**입니다.
문제 해결 시, 입력 크기와 성격에 따라 적절한 컨테이너를 선택하는 것이 효율에 직접적인 영향을 미칩니다.

자료구조	특징	정렬과의 관계
vector	동적 배열, 연속 메모리 구조, 랜덤 접근 O(1)	대부분의 정렬 문제에서 기본 사용
set	자동 정렬 + 중복 불가	간단한 오름차순 정렬 문제(예: 백준 2750번)에 유용 단, vector에 비해 속도가 느림
map	Key 기반 자동 정렬, Key 중복 불가	빈도 계산 및 통계 문제에서 활용
string	문자 단위 처리, 비교 연산 지원	사전순 정렬 문제에서 활용

2. STL 정렬 관련 주요 함수 및 기반 알고리즘

함수	내부 알고리즘	사용 목적 / 상황	특징
sort()	**IntroSort (QuickSort + HeapSort + InsertionSort)**	일반 정렬	평균 O(N log N), 불안정 정렬
stable_sort()	**MergeSort 기반**	동일 키의 순서 유지 필요 시	O(N log N), 안정 정렬

partial_sort()	**HeapSort 기반**	상위/하위 일부 원소만 필요할 때	O(N log K)
nth_element()	**QuickSelect 기반**	K번째 원소만 필요할 때	평균 O(N)
max_element()	선형 탐색	최대값 탐색	O(N)

활용 예시

- stable_sort → 백준 10814(나이순 정렬): 입력 순서 유지 필요
- nth_element → 백준 11004(K번째 수): 정렬 없이 K번째 원소 탐색
- partial_sort → 백준 2108(통계학): 범위, 최댓값/최솟값 빠른 추출

3. 플랫폼별 문제 접근 차이

구분	**프로그래머스**	**백준**
중심 포인트	문제 **해석력** 중심	**알고리즘 효율성** 중심
출력 형식	대부분 함수 반환형	콘솔 출력 중심
문제 유형	STL 함수 활용 중심	입력 최적화 + 알고리즘 선택 중심
대표 예시	문자열 분리 및 사전순 정렬	수 정렬하기 1, 2, 3 (입력 크기별 알고리즘 적용)

※ 결론:
프로그래머스는 "문제를 정확히 해석하고 STL로 간결하게 구현"하는 연습에 좋고, 백준은 "입력 최적화와 알고리즘 효율"을 검증하는 훈련에 적합합니다.

4. 입력/출력 최적화 및 기타 팁

항목	**설명**	**비고**
ios::sync_with_stdio(false);	C의 stdio와 동기화 해제 → 입력속도 향상	필수
cin.tie(nullptr);	cin, cout 묶임 해제 → 출력 지연 제거	필수

'₩n' vs endl	endl은 flush 포함되어 느림	'₩n' 권장
auto	타입 추론 자동화로 코드 간결화	C++11 이상
stringstream	문자열 파싱에 유용	공백 분리 등
범위형 for문	for (auto &x : v) 형태	간결 + 안전

5. 백준 문제 요약 (핵심 학습 포인트)

문제	핵심 알고리즘	비고
2750 수 정렬하기	set 또는 sort	기본 정렬
2751 수 정렬하기 2	vector + sort	대량 데이터 (입력 최적화 필수)
10989 수 정렬하기 3	**계수 정렬 (Counting Sort)**	메모리 최적화
11650 좌표 정렬하기	sort + custom compare	다중 기준 정렬
10814 나이순 정렬	stable_sort	입력 순서 보존
1181 단어 정렬	unique + sort	중복제거 + 길이순
11004 K번째 수	nth_element	부분정렬
2108 통계학	round, max_element, nth_element	통계값 계산

6. 프로그래머스 문제 요약 (핵심 학습 포인트)

문제	포인트	사용 함수
문자열 자르기	"x" 기준 split → 사전순 정렬	sort, stringstream
소문자 정렬	tolower, 문자열 정렬	sort
기준값 가까운 순서	커스텀 비교	sort + lambda
배열 자르기	범위 슬라이스 + 정렬	sort

문자열 n번째 기준	문자 비교 + 사전순	sort
H-Index	내림차순 정렬 + 인덱스 비교	sort, greater<>
가장 큰 수 만들기	문자열 비교 조합	sort + custom compare
파일명 정렬	HEAD/NUMBER 추출 후 정렬	stable_sort

7. 학습을 통해 느낀 점

1. **STL 정렬 함수들은 단순히 정렬을 수행하는 도구가 아니라**, 내부 알고리즘과 안정성(stability)에 따라 활용 전략이 달라진다.
2. **문제 풀이에서는 "정렬 기준"을 명확히 정의하는 것이 절반이다.**
   (예: 다중 조건, 음수 보정 offset, 중복 제거 등)
3. **입력 최적화** (ios::sync_with_stdio, cin.tie)는 대규모 입력 시 성능 차이를 극명하게 만든다.
4. **문제별 데이터 크기와 특성을 고려한 알고리즘 선택**이 핵심이다.
   - 작은 입력: sort, set
   - 큰 입력: counting sort, nth_element
5. STL 자료구조와 알고리즘을 **함께 병행 학습**해야 실전 코테 문제에 유연하게 대응할 수 있다.